CBM 64 USER MANUAL ERRATA

Page vii INTRODUCTION, paragraph 3

... the sprite editor lets you animate as many as 8 different picture levels at one time.

... at one time. The Sprite Editor is available from your Commodore dealer as a program that loads directly into the Commodore 64.

Page 2 SIDE PANEL CONNECTIONS, Item 3. Game ports

Each game connector can accept a joystick, game paddle, or a light pen.

... can accept a joystick or game paddle. The game port closest to the front of the computer can accept a light pen.

Page 4 CONNECTIONS TO YOUR TV, Item 1.

... numbered 6 in the diagram. Only one end of the cable will fit in here.

... numbered 6 in the diagram. Either end of the cable will fit in here.

Page 7 USING THE COMMODORE 64, Item 1.

... the rocker switch on the left-side panel.

... the rocker switch which is on the right side panel as you look from the front.

Page 9 TROUBLESHOOTING CHART, Item 4.

Picture with excess background noise

Sound with excess background noise.

Page 9 CURSOR

... flashing square next to READY ...

... flashing square underneath READY ...

Page 34 EDITING TIPS, last paragraph.

Try editing our sample program ..... and adding a semicolon to the end of the line ...

... our sample program on page 33 ..... and adding a comma to the end of the line.

Page 45 INPUT, last line.

prompt must be less than 40 characters

prompt must be 38 characters or less

Page 51 PROGRAM, line 60.

IF GU = NM THEN PRINT "GREAT! YOU GOT MY NUMBER"

PRINT "GREAT! YOU GOT MY NUMBER"

Page 64 SCREEN MEMORY MAP, top of page.

POKE 1524,81

POKE 1524,81:POKE 1524+54272,1

Page 76 ADDITIONAL NOTES ON SPRITES, an additional paragraph.

To define more than 2 sprites you will need additional blocks of memory in which to store your sprite data. You can use some of the RAM reserved for BASIC by moving the start of BASIC up in memory. To do this type the following before loading your program.

POKE44,16:POKE16*256,0:NEW

You can now use blocks 32 through 41, (locations 2048 - 4095) for your sprite data without disturbing any special locations that are used by the computer.

Page 77 BYTE, second sentence.

... you can actually have a total of 255 different combinations.

... you can actually have a total of 256 different combinations.

Page 142 Cartridge Expansion Slot Changes to:

Pin	Type	Pin	Type
11	ROML	1	GND
10	I/O2	2	+ 5V
9	EXROM	3	+ 5V
8	GAME	4	IRQ
7	I/O1	5	R/W
6	Dot Clock	6	Dot Clock
5	CR/W	7	I/O1
4	IRQ	8	GAME
3	+ 5V	9	EXROM
2	+ 5V	10	I/O2
1	GND	11	ROML
Z	GND	N	A9
Y	CA0	P	A8
X	CA1	R	A7
W	CA2	S	A6
V	CA3	T	A5
U	CA4	U	A4
T	CA5	V	A3
S	CA6	W	A2
R	CA7	X	A1
P	CA8	Y	A0
N	CA9	Z	GND
M	CA10	A	GND
L	CA11	B	ROMH
K	CA12	C	RESET
J	CA13	D	NMI
H	CA14	E	SO2
F	CA15	F	A15
E	SO2	H	A14
D	NMI	J	A13
C	RESET	K	A12
B	ROMH	L	A11
A	GND	M	A10

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
A	B	C	D	E	G	H	J	K	L	M	N	P	R	S	T	U	V	W	X	Y	Z

Changes to:

22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Z	Y	X	W	V	U	T	S	R	P	N	M	L	K	J	H	G	E	D	C	B	A

Page 147 PROGRAM, line 220.

```
K$ = "Q2W3ER5T6Y7UI9OOP@-*  "
```

```
K$ = "Q2W3ER5T6Y7UI9OOP@-*+£  "
```

Page 150 FORMULA TOO COMPLEX

... to work with.

... to work with or a formula has too many parentheses.

Page 154 MUSIC NOTE VALUES, additional paragraph.

Filter Settings

Location	Contents	
54293	low cutoff frequency	(0-7)
54294	high cutoff frequency	(0-255)
54295	resonance	(bits 4-7)
	filter voice 3	(bit 2)
	filter voice 2	(bit 1)
	filter voice 1	(bit 0)
54296	high pass	(bit 6)
	band pass	(bit 5)
	low pass	(bit 4)
	volume	(bits 0-3)

Page 157 APPENDIX O, Sprite Register Map

17 11 RC8 EC5 BSM BLNK RSEL YSCL2 YSCL1 YSCLO

17 11 RC8 ECM BMM BLNK RSEL YSCL2 YSCL1 YSCLO Y SCROLL/MODE

Page 158 APPENDIX O, Sprite Register Map

22 16 N.C. N.C. RST MCM CSEL XSCL2 XSCL1 XSCLO

22 16 N.C. N.C. RST MCM CSEL XSCL2 XSCL1 XSCLO X SCROLL/MODE

24 18 VS13 VS12 VS11 CB13 CB12 CB11 CB10 N.C. SCREEN char Mem

24 18 VS13 VS12 VS11 VS10 CB13 CB12 CB11 N.C. SCREEN char Mem

Page 161 VOICE NUMBER 2, WAVEFORM POKE

54288

54283

QUICK REFERENCE CARD, SYSTEM COMMANDS

WAIT X,Y,Z Program waits until contents of location X, when FORed with Z and ANDed with Y, is nonzero.

WAIT X,Y,Z Program waits until contents of location X, when EORed with Z and ANDed with Y, is nonzero.

Detecting Function Keys

The function keys on the Commodore 64 are a set of four spare keys located on the right hand side of the keyboard. When using a basic program it is often desirable to display a message such as:

"Press F1 to continue"

The Commodore 64 makes it possible to detect when a function key has been pressed because each of these keys has a CHR$ code assigned to it. These codes range from 133 to 140 for keys F1 through to F8. A simple way to wait for the F1 function key to be pressed is as follows:

```
100 GET A$: IF A$<>CHR$(133) THEN 100
```

However, if you do not want to use CHR$ every time you want to check for a function key, then you can use the following method. If you type a single quote and then press each of the function keys in turn, you will see a series of reversed video graphics appear. We can use this property of the function keys to make the test loop a little easier to key in, as follows:

```
100  GET A$: IF A$<>"[F1]" THEN 100
```

Please note that the 'F1' in square brackets means press that key, not type it as shown. The F1 key is shown as a solid square with a line through the top of it between the quotes.

Another way of testing for the function keys (and any other keys for that matter) is to PEEK memory location 197 which contains a special code that tells the computer which row and column of the keyboard the key being pressed is located. This location contains 64 when no keys are depressed and a different number if a key is held down. We can use this facility to monitor the function keys all of the time and even wait for it to be released before continuing with the rest of the program. The following codes are put into location 197 when each of the function keys are pressed:

F1 = 4
F2 = 5
F3 = 6
F4 = 3

For example:

```
10 IF PEEK(197)<>4 THEN 10: REM WAIT FOR F1
20 IF PEEK(197)<>64 THEN 20:REM WAIT FOR RELEASE
```

will wait for the F1 key to be pressed and then released before continuing with the rest of the program.

USER DEFINED FUNCTION KEYS

The BASIC program on the following page will allow you to define the function keys so that they print anything you like up to eight characters long.

Lines 200 - 330 poke the machine code into $C000 onwards. If the program stops at line 320 and prints the error message, then you have made a mistake in typing the data and should go through again.and check it very carefully. If the data is poked in successfully then you will be able to use the function keys as if each one were a single key with a word on it.

```
0   REM FUNCTION KEY DEFINE BY S BEATS
1   GOSUB 300                        : REM POKE IN DATA
2   F$(1) = "LIST" + CHR$(13)        : REM YOU CAN CHANGE THESE
3   F$(2) = "RUN" + CHR$(13)         : REM DEFINITIONS IF YOU WANT
4   F$(3) = "GOSUB"                  : REM BUT REMEMBER IT IS
5   F$(4) = "SAVE" + CHR$(34)        : REM 8 CHARACTERS MAXIMUM
6   F$(5) = "LOAD"                   : REM PER KEY
7   F$(6) = "GOTO"
8   F$(7) = "CHR$("
9   F$(8) = "RETURN"
10  SYS 12 * 4096 + 64               : REM ENABLE CODE
20  V = 12 * 4096 - 1                : REM START OF DATA
30  FOR I = 1 TO 8: K = I - 1: V1 = V + K * 8
40  FOR J = 1 TO LEN(F$(I)           : REM POKE IN KEY DEFINITIONS
50  POKE V1 + J, ASC(MID$(F$(I),J,1))
60  NEXT: NEXT: END
200 DATA 120,169,87,141,20,3,69,192,141,21
210 DATA 3,88,162,63,169,0,157,0,192,202
220 DATA 16,250,96,165,197,201,64,208,6,141
230 DATA 151,192,76,148,192,205,151,192,240,44
240 DATA 141,151,192,162,3,221,152,192,240,5
250 DATA 202,16,248,48,29,138,174,141,2,240
260 DATA 2,24,105,4,10,10,10,168,162,0
270 DATA 185,0,192,157,119,2,200,232,224,8
280 DATA 208,244,134,198,76,49,234,64,4,5,6,3,0
300 FOR I = 0 TO 92: READ A: Z = Z + A
310 POKE 12 * 4096 + 64 + I,A: NEXT
320 IF Z <> 10771 THEN PRINT "ERROR IN DATA STATEMENTS": STOP
330 RETURN
```